Université de Montréal
C.P. 6128, Succursale Centre-ville,
Montréal (Québec) Canada H3C 3J7

Téléphone : +1-514-343-7575
Télécopieur (Fax) : +1-514-343-7121
crt@crt.umontreal.ca
www.crt.umontreal.ca/CRT/

CRT-2001-10

Centre de recherche sur les transports (C.R.T.)
*Centre for Research on Transportation*

# *HEURISTICS FOR THE TRAVELING PURCHASER PROBLEM*

by

**Fayez F. Boctor**
**Gilbert Laporte**
**Jacques Renaud**

**Mars 2001**

102-03123/

**PUBLICATION**
**CRT-2001-10**

# *HEURISTICS FOR THE TRAVELING PURCHASER PROBLEM**

**by**

**Fayez F. Boctor**[1,2]
**Gilbert Laporte**[3]
**Jacques Renaud**[2,4]

* This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged.

1 Faculté des sciences de l'administration, Université Laval, Québec, Canada G1K 7P4
2 Centre de recherche sur les technologies de l'organisation réseau, Université Laval, Québec, Canada G1K 7P4
3 École des Hautes Études Commerciales de Montréal, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 1V6 et Centre de recherche sur les transports, C.P. 6128, succ. Centre-ville, Montréal, Canada H3C 3J7
4 Télé-Université, 455 rue de l'Église, C.P. 4800, succ. Terminus, Québec, Canada G1K 9H5

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2001

## ABSTRACT

This article deals with two versions of the traveling purchaser problem. In the uncapacitated version, the number of units of a given product available any market where it is sold is either larger than or equal to the demand. In the capacitated version, the availability may be smaller than the demand. This study extends some known heuristics and presents some new ones capable of solving either version of the problem. The new heuristics are compared to each other and to some previous heuristics. Computational results confirm the quality of the proposed heuristics.



**Keywords** : Routing, traveling purchaser problem, perturbation heuristics



## RÉSUMÉ

Cet article traite de deux versions du *problème de l'acheteur itinérant.* Dans la version sans limite de capacité, le nombre d'unités d'un produit disponibles à un marché où ce produit est vendu est supérieur ou égal à la demande. Dans la version avec capacité, le nombre d'unités d'un produit à un marché peut être inférieur à sa demande. Cet article adapte quelques heuristiques connues et de nouvelles heuristiques de perturbation pour les deux versions du problème. Des tests comparatifs montrent que les heuristiques proposées produisent d'excellents résultats.



**Mots clefs** : Problèmes de tournées, problème de l'acheteur itinérant, heuristiques de perturbation.

## 1- INTRODUCTION

An interesting generalization of the well-known *Traveling Salesman Problem* (TSP) is the *Traveling Purchaser Problem* (TPP) first introduced by Ramesh (1981). The undirected version of this problem can be stated as follows. Consider a domicile denoted by 0, a set of markets denoted by $M=\{1, 2, \ldots, m\}$, a travel cost $c_{ij}$ on each edge $(i, j)$ linking two markets, and a set $K=\{1, 2, \ldots, n\}$ of products. Denote by $M_k$ the set of markets selling product $k$ and by $p_{ki}$ the price of product $k$ at market $i$. In what follows, $c_{ij}$ must be interpreted as $c_{ji}$ whenever $i>j$. The TPP is to construct a tour through a subset of the $m$ markets and the domicile and to purchase each of the $n$ products at one of these markets so as to minimize the sum of the travel and purchase costs. Under Ramesh's definition, it is implicitly assumed that if a product is available at a given market, its quantity is sufficient to satisfy the demand. This version of the problem will be called the *Uncapacitated Traveling Purchaser Problem* (UTPP). Recently, Laporte Riera-Ledesma and Salazar-González (2000) have solved a generalization of the UTPP where the demand for product $k$ is $d_k$, and the availability $q_{ki}$ of product $k$ at market $i$ may be less than $d_k$. This version will be called the *Capacitated Traveling Purchaser Problem* (CTPP). It is convenient to define the UTPP as a special case of the CTPP where $d_k=1$ for all $k$ and $q_{ki}=0$ or 1 for all $k$ and $i$. In both versions, the traveling purchaser may visit any market as many times as necessary, but since we assume that travel costs satisfy the triangle inequality, there always exists an optimal solution in which each market is visited exactly once. Some authors, e.g., Laporte *et al.* (2000), have considered a version of the TPP in which the triangle inequality is not assumed, but still impose a maximum of one visit at each market.

The most common TPP applications occur in vehicle routing and warehousing (Singh and Van Oudheusden, 1997). An interesting application in the field of production scheduling is also described by Buzacott and Dutta (1971). Here, a multi-purpose machine can assume several configuration $i$ and each task $k \in K$ must be performed using a configuration in a set $M_k$. Travel costs $c_{ij}$ correspond to changeover costs between jobs.

The TPP is NP-hard since it reduces to the TSP if each product is available only at one market and each market sells only one product. The TPP also reduces to the *Uncapacitated*

*Facility Location Problem* (UFLP) if $c_{ij}$ is equal to the average opening cost of locations $i$ and $j$, and $p_{ki}$ is the cost of serving customer $k$ from facility $i$. The UTPP can also be transformed into a *Generalized Traveling Salesman Problem* (GTSP). In this problem the sites (customers or markets) are partitioned into clusters and the traveling purchaser must visit at least one site per cluster (see, e.g., Renaud and Boctor 1998). Again, if travel costs satisfy the triangle inequality, it is not optimal to visit more than one site in a given cluster. To transform a TPP into a GTSP, each market is replicated into as many counters as the number of different products it offers, and each counter sells only one product. The GTSP is obtained by defining a cluster as the set of all counters selling the same product. Half the price of each product is then added to the cost of the edges incidents to the counter. Unfortunately, this transformation does not yield any computational advantage for the solution of the TPP.

The TPP has received the attention of several researchers, most of whom have proposed heuristics for its solution. To our knowledge the only available exact algorithms are the lexicographic algorithm of Ramesh (1981), the branch-and-bound algorithm of Singh and Van Oudheusden (1997), and the branch-and-cut algorithm of Laporte *et al* (2000). Our purpose is to develop new and more powerful heuristics for the TPP. In a series of tests, we show that they outperform all previously published heuristics. By comparing our results to the exact solution values obtained on the same instances by Laporte *et al* (2000), we are also able to prove that our heuristics consistently yield near-optimal solutions.

The remainder of this article is organized as follows. We provide in Section 2 a review of all known algorithms for the TPP. A new implementation of one of these methods is described in Section 3. The new heuristics are described in Section 4 followed by computational results in Section 5 and by conclusions in Section 6.

## 2- LITERATURE REVIEW

One of the earliest algorithms for the UTPP is the lexicographic implicit enumeration method developed by Ramesh (1981). While this algorithm is exact in principle, it only works on very small instances. A heuristic can be derived from it by truncating the search at the first feasible solution. Pearn and Chien (1998) have tested this heuristic, called the *Lexicographic*

*Search Heuristic* (LSH) as well as two improved versions that they called *Next-Block Search Heuristic* (NB-SH) and *Next-Neighbor Search Heuristic* (NN-SH).

Golden, Levy and Dahl (1981) have proposed the following *Generalized Savings Heuristic* (GSH) for the UTPP. It starts with an initial tour containing the domicile and the market selling the largest number of products at their lowest available price. Ties are broken by selecting the market with the smallest sum of product prices. At each iteration, the unvisited market producing the largest cost saving is inserted in the current tour. The heuristic stops when no more savings can be achieved. Pearn and Chien (1998) have suggested two improved versions of this heuristic. The first one, called the *Parameter-Selection Generalized-Savings Heuristic* (PS-GSH), uses a weighted saving function where a term reflecting the purchase saving at a given market is multiplied by a weight $\theta$ and added to the travel cost saving. The PS-GSH repeatedly solves the instance with seven different values of $\theta$. The second version, called the *Tie-Selection Generalized-Savings Heuristic* (TS-GSH), is similar to the original heuristic, but the tie-breaking rule selects the market closest to the domicile instead of the market offering the smallest sum of product prices.

The following *Tour Reduction Heuristic* (TRH) was suggested by Ong (1982) for the UTPP. It starts with an initial tour containing a subset of markets offering the $n$ products and iteratively drops the market yielding the largest cost reduction until no further improvement can be obtained. Ong also suggested using a good TSP algorithm to resequence the markets in the intermediate tours. Obviously, the performance of the TRH heavily depends on the initial subset of markets, on the number of times the TSP heuristic is applied, and on the performance of the TSP heuristic. Pearn and Chien (1998) suggested initially selecting the set $C$ of markets selling at least one product at its lowest price. They also tested two variants of the TRH. In the first one, called *Adjusted-Cheapest Tour-Reduction Heuristic* (AC-TRH) the initial set of markets contains, in addition to $C$, all markets for which the price of one or more products augmented by their travel cost to the domicile is minimal. In the second variant, called *Nearest-Cheapest Tour-Reduction Heuristic* (NC-TRH) the initial set includes, in addition to $C$, the closest $q$ markets to the domicile. They solved the TPP with five different values of $q$.

Another heuristic proposed by Pearn and Chien (1998) is called the *Commodity Adding Heuristic* (CAH). This heuristic implicitly assumes that all products are available at all markets. The procedure considers the first product from a list and constructs a least cost solution for this product. At each following iteration, it inserts the next product in the solution in a least cost manner. In some cases, the best market for this product may already be in the solution; in other cases, it has to be inserted. The authors also propose improving the solution by means of the Basart and Huguet (1989) TSP heuristic, or by market drop or market interchange operations.

More recently, Laporte *et al* (2000) have developed a *Market Adding Heuristic* (MAH) applicable to both the UTPP and the CTPP. It gradually extends a cycle by inserting at each step a new market selling a product whose demand is not yet fully satisfied. More specifically, the procedure determines in which market each such product is available at the lowest price and it first inserts in the tour the market corresponding to the highest price. The insertion of this market in the current tour is made accordingly to a least insertion cost rule. Once a feasible cycle has been obtained, it is post-optimized by iteratively acting on the set of markets in the solution, the assignment of products to markets, and the routing cost over the visited markets.

Finally, post-optimization procedures based on simulated annealing and on tabu search were proposed by Voß (1996). Simulated annealing seems to work best for the CTPP, while tabu search is recommended for the UTPP.

In addition to the lexicographic method of Ramesh (1981), two exact approaches are available. Singh and Van Oudheusden (1997) have developed a branch-and-bound algorithm for the UTPP capable of solving instances of up to 25 markets and 100 products. More recently, Laporte *et al* (2000) have proposed a branch-and-cut algorithm for both the UTPP and the CTPP. At the root of the search tree, the choice of variables included in the first linear program (LP) is guided by the MAH solution. This LP is solved and valid inequalities are introduced according to the usual branch-and-cut rules. The final LP solution obtained at the root of the search tree is then used as a basis for an *LP Based Market Adding Heuristic* (LPMAH). This consists of applying the MAH starting with edges whose associated variables have the largest values. This branch-and-cut algorithm was successfully applied to instances involving up to 200 markets and 200 products.

## 3- NEW COMMODITY ADDING IMPLEMENTATIONS

The *Commodity Adding Heuristic* (CAH) described by Pearn and Chien (1998) applies only to the UTPP and some of its components are not fully explicited in the original description. Since this appears to be one of the best heuristics for the UTPP, we have produced our own implementation and tested a number of variants. Our version of the CAH applies to both the CTPP and the UTPP. We consider a CTPP implementation which can also be used for the UTPP by appropriately defining the input data. The algorithm contains a construction phase followed by an improvement phase.

***Construction phase***

*Step 1* (*Initialization*). Randomly generate a list of all products. Relabel the products according to their order in the list. Set $h$:=1.

*Step 2* (*First units of product 1*). Determine the market $i^*$ for which the unit purchase cost of product 1 is minimized:

$$i^* := \arg\min_{i \in M_1} \{2c_{0i}/q_{1i} + p_{1i}\}. \qquad (1)$$

Purchase all available units of each product at market $i^*$ without exceeding their unfulfilled demand. The initial partial tour is given by $T$:={0,$i^*$,0}.

*Step 3* (*Remaining units of product 1*). Let $\overline{M}_h$ be the set of unvisited markets where product $h$ is available and identify the market $i^*$ yielding:

$$i^* := \arg\min_{i \in \overline{M}_1} \{(R(i,T) + P(i,T)\}, \qquad (2)$$

where $R(i,T)$ is the minimum insertion travel cost of market $i$ in tour $T$, and $P(i,T)$ is the minimum total purchase cost of products to be purchased at $T \cup \{i\}$. Insert $i^*$ in $T$ so as to minimize the insertion cost. Let $z^*$ be the cost of the current partial solution. If all the required units of product 1 are purchased, go to Step 4; otherwise repeat Step 3.

*Step 4* (*Termination test*). If product $h$ has not been fully purchased or if some units of product $h$ can be purchased at a lower cost in a market belonging to $M \backslash T$, go to Step 5. Otherwise set $h$:=$h$+1. If $h>n$, go to the improvement phase; otherwise repeat Step 4.

*Step 5* (*Purchasing more units of h or buying h at a lower price*). Identify the market $i^*$ yielding:

$$i^* := \arg\min_{i \in \bar{M}_h} \{R(i,T) + P(i,T)\}. \quad (3)$$

If product $h$ is not yet fully purchased, or if it is fully purchased and $R(i^*,T)+P(i^*,T)<z^*$, then insert $i^*$ in $T$ to minimize the insertion cost, update $z^*$ and go to Step 4.

***Improvement phase***

*Repeat Steps 6 to 9 until no further improvement can be obtained*

*Step 6* (*Market drop*). Consider in turn each visited market and remove it if this yields a total cost reduction.

*Step 7* (*Market add*). Consider in turn each unvisited market. Insert it in the current tour $T$ so as to minimize the added travel cost. Reallocate all products to visited markets to minimize the purchase cost. Drop from the solution all markets where no purchase is made. Retain the resulting tour if it yields a total cost reduction.

*Step 8* (*Market exchange*). Perform the following operations until no further improvement can be obtained. Drop a market from the tour and add to the resulting tour an unvisited one as in Step 7. If the resulting tour is feasible and less costly, implement the exchange move.

*Step 9* (*TSP heuristic*). Attempt to shorten the current tour by means of a TSP heuristic. In our implementation we use the $I^3$ heuristic of Renaud, Boctor and Laporte (1996).

This heuristic works equally well for the UTPP and CTPP. In the UTPP implementation Step 3 is never entered and Step 5 is executed exactly once. Two versions have been tested. In the first version, called CAH1, the construction phase is used to generate ten solutions and the improvement phase is only applied to the best one. In the second version, called CAH2, both the construction and improvement phases are applied starting from each of ten different orderings of the products in Step 1, and the best of all available solutions is retained.

## 4- PERTURBATION HEURISTICS

We have developed three *Perturbation Heuristics* (PH) for the TPP. These are post-optimization schemes in which an improvement procedure is applied to a *perturbed* solution, as it is commonly done in genetic search (Dowsland, 1996), for example. The interest of working with a perturbed solution is to help the search process escape from a local minimum. For a recent

article on perturbation heuristics, see Renaud, Boctor and Laporte (2000). Two of the PH we have developed apply to the UTPP and will be called UPH1 and UPH2. The third one applied to the CTPP and is called CPH.

The three PH combine in different ways seven basic procedures: 1) Market drop; 2) Market add; 3) Market exchange; 4) TSP heuristic; 5) Cheapest insertion; 6) Double market drop; 7) Double market exchange. The first four procedures correspond to Steps 6, 7, 8 and 9 of Section 3, respectively. We now describe the remaining three basic procedures followed by UPH1, UPH2 and CPH.

**Cheapest insertion**

The *Cheapest Insertion Heuristic* (CIH) applied to the UTPP and to the CTPP. It is applied ten times with different values of a parameter $\alpha \in (0,2)$ controlling the relative weight of travel cost and purchase cost.

*Step 1* (*Initial selection of markets*). Determine an initial set of markets $M_0$ to be included in a starting tour. In the CTPP this set is defined as

$$M_0 := \{0\} \cup \left\{ i \in M : \text{there exists } k \in K \text{ such that } \sum_{j \in M_i \setminus \{i\}} q_{kj} < d_k \right\}.$$

In other words, $M_0$ includes all markets that must necessarily belong to any feasible solution. If $M_0 = \{0\}$ then $M_0$ is identified as for the UTPP. In the UTPP, $M_0 := \{0, i^*, j^*\}$, where $i^*, j^*$ are two markets yielding

$$\underset{i,j \in M}{Min} \{\alpha(c_{0i} + c_{0j} + c_{ij}) + (2-\alpha)P(i,j)\}$$

where $P(i,j)$ is the most economical purchase price of all available products at markets $i$ and $j$ without exceeding their demand.

*Step 2 (Initial tour)*. Determine an ordered set $T$ corresponding to a tour on $M_0$ by means of the TSP heuristic. If the solution is feasible, set $\delta := 1$; otherwise set $\delta := 0$.

*Step 3 (Market selection)*. Determine a market $l^*$ to be considered for inclusion in the tour, yielding

$$s^* = \underset{l \in M \setminus T,\ i,j \in T}{Min} \{\alpha(c_{il} + c_{lj} - c_{ij}) + (2-\alpha)S(l,T)\}$$

where $S(l,T)$ is the purchase saving achieved if market $l$ is included in the solution, all products are optimally purchased at the available markets, and at least one product is purchased at $l$. This saving is unrestricted in sign since adding market $l$ to the solution may mean that additional products can now be purchased.

*Step 4 (Insertion test).* If $\delta=0$, insert $l^*$ in $T$ to minimize the insertion cost; if the solution becomes feasible, set $\delta:=1$; go to Step 3. If $\delta=1$ and $s^*<0$, insert $l^*$ in $T$ to minimize the insertion cost and go to Step 3. If $\delta=1$ and $s^*\geq 0$, go to Step 5.

*Step 5 (Improvement).* Attempt to improve the current solution (if it differs from a previously solution) by applying Market drop, Market add and TSP heuristic.

*Step 6 (Combination).* If the current solution $T$ differs from the best known solution $T^*$, attempt to create a better solution $\overline{T}$ by combining $T$ and $T^*$. More specifically, all markets present in $T^*$ but not in $T$ are gradually inserted in the tour $T$ using a cheapest insertion criterion. Step 5 is then applied to $\overline{T}$.

In this procedure, the role of $\alpha$ is to properly adjust the relative weight put on routing costs and on purchase cost. We have indeed observed that one of these costs often predominates in a given instance, but this is not known in advance. By using ten different values of $\alpha$ (0.4, 0.5, ..., 1.3), one is almost certain to arrive at the right ponderation. Another observation relates to the potential selection of market $l$ in Step 3. It is often the case that a market $l$ yielding $s^*$ is in fact useless since no purchases are ever made at that market and its presence in the solution only increases the travel cost. This is why such markets are only considered if they are used to purchase at least one product.

**Double market drop**

Consider each pair of visited markets and remove the pair that yields the largest cost reduction. Repeat until no more cost reduction can be achieved.

**Double market exchange**

This procedure is similar to Market exchange except that *two consecutive* markets are dropped from the tour and *one* market is added.

**Perturbation heuristics UPH1, UPH2 and CPH**

We first describe UPH1 and we then indicate how this procedure is modified to yield UPH2 and CPH.

*Step 1 (Initial tour).* Construct and initial solution by applying Cheapest insertion, Market drop, Market exchange, Market Add and TSP heuristic.

*Apply Step 2 and 3 until no improvements are obtained for $\beta$ successive iterations*

*Step 2 (Construction of a perturbed solution).* Eliminate up to $\gamma$% of all market from the solution. Gradually reinsert a randomly selected unvisited market in the solution using a cheapest insertion criterion, until feasibility is reached.

*Step 3 (Improvement).* Attempt to improve the current solution by applying Market drop, Market exchange, Market add, TSP heuristic, Double market exchange, and Double market drop.

*Step 4 (Intensification).* Successively apply Market add, Market exchange, Market drop and TSP heuristic to the best known solution until no further improvement is obtained. Then apply Double market exchange and Double market drop to the best known solution.

In our implementation we use $\beta$=15 and $\gamma$=50.

Procedure UPH2 is similar except that Step 6 of the Cheapest insertion procedure is applied after Step 3 and also repeated $\beta$ times. Procedure CPH is also similar to UPH1 except that Double market exchange and Double market drop are not applied in Step 3. Applying these procedures would have been too time consuming.

## 5- COMPUTATIONAL RESULTS

The procedures just described were coded in Borland Delphi 3.0 and run on a PC Celeron 500 Mhz under Windows 95 with a maximum running time of 3600 seconds for any given instance. These were tested on both the uncapacitated and the capacitated instances used by Laporte *et al.* (2000) and available on the http://webpages.ull.es/users/jjsalaza web site. These instances range from $m$=50 to 350 markets and from $n$=50 to 200 products in the uncapacitated case. Five instances were generated for each combination of $m$ and $n$. However, the optimal

solution is known for only 80 of these instances out of 140; in the remaining 60 cases, the best known solution value is used in all comparisons. Accordingly, we have divided these 140 instances in two groups: Uncapacitated (optimal) and Uncapacitated (best). Similarly, capacitated instances range from $m$=50 to 350 markets and from $n$=50 to 200 products, with seven values of a parameter $\lambda$ controlling the demand on a product $k$. In these instances, $q_{ki}$ is randomly generated in [1,15] and $d_k := \lambda \underset{i \in M_k}{\text{Max}}\, q_{ki} + (1-\lambda) \sum_{i \in M_k} q_{ki}$ where $\lambda$=0.1, 0.5, 0.7, 0.8, 0.9, 0.95 and 0.99. Again five instances were generated for each combination of $m$, $n$ and $\lambda$ yielding 980 instances, 439 of which having a known optimal solution. Again these instances are divided between Capacitated (optimal) and Capacitated (best).

Average computational results are presented in Tables 1 to 4. The column headings are as follow:

\# : number of instances over which the average is computed;

LPMAH : the LP based MAH implemented by Laporte *et al.* (2000) (computation times for this heuristic only were obtained on a Pentium 500 Mhz computer running Linux with a C++ code);

MAH : our implementation of the Laporte *et al.* (2000) Market Adding Heuristic;

CAH1 : our first implementation of the Commodity Adding Heuristic;

CAH2 : our second implementation of the Commodity Adding Heuristic;

UPH1 : our first Uncapacitated Perturbation Heuristic;

UPH2 : our second Uncapacitated Perturbation Heuristic;

CPH : our Capacitated Perturbation Heuristic;

% : percentage deviation of the heuristic solution value over the optimal value (Tables 1 and 3); percentage deviation of the heuristic solution value over the best known solution value (Tables 2 and 4); the best known solution was produced by one of the heuristics used in the comparison and sometimes in a preliminary version developed during the development phase;

Seconds : computation time in seconds.

Computational results presented in Tables 1 and 2 indicate that both UPH1 and UPH2 produce solution values within 0.75% of the optimum for $m \leq 200$ and typically close to 1% of the best known solution value for $250 \leq m \leq 350$, which is a clear indication of the quality of these algorithms. The proposed heuristics also produce smaller optimality gaps than any other heuristic used in the comparison. When comparisons are made on instances for which an optimal solution is known, the gaps tend to increase with the number of markets but are not clearly related to the number of products. Computation times increase steadily with the number of markets and the number of products. On the whole, the running times of UPH1 and UPH2 are similar for $m \leq 200$, but UPH1 tends to be faster for $250 \leq m \leq 350$. Both these heuristics are also faster than LPMAH and CAH2. On these instances the MAH behaves rather poorly.

**Table 1. Average computational results, uncapacitated (optimal) instances**

| Heuristic | # | LPMAH | | MAH | | CAH1 | | CAH2 | | UPH1 | | UPH2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | % | Seconds | % | Seconds | % | Seconds | % | Seconds | % | Seconds | % | Seconds |
| $m$=50 | 20 | 1.57 | 22 | 11.08 | 2 | 3.05 | 1 | 0.53 | 12 | 0.33 | 12 | 0.26 | 12 |
| $m$=100 | 20 | 2.43 | 260 | 10.47 | 22 | 1.89 | 9 | 0.64 | 104 | 0.28 | 60 | 0.30 | 50 |
| $m$=150 | 20 | 7.07 | 1482 | 24.14 | 96 | 2.12 | 20 | 0.97 | 263 | 0.59 | 213 | 0.71 | 184 |
| $m$=200 | 18 | 9.61 | 3225 | 24.63 | 165 | 2.59 | 34 | 0.95 | 513 | 0.65 | 261 | 0.53 | 324 |
| $m$=250 | 11 | 10.20 | 5254 | 29.67 | 249 | 2.66 | 38 | 1.04 | 649 | 0.47 | 233 | 0.63 | 253 |
| $n$=50 | 25 | 3.84 | 918 | 21.46 | 57 | 0.78 | 6 | 0.43 | 111 | 0.34 | 17 | 0.35 | 16 |
| $n$=100 | 22 | 5.74 | 2008 | 19.19 | 81 | 2.03 | 17 | 0.73 | 216 | 0.49 | 74 | 0.59 | 71 |
| $n$=150 | 22 | 6.09 | 1925 | 19.93 | 118 | 3.14 | 23 | 1.05 | 371 | 0.48 | 191 | 0.45 | 225 |
| $n$=200 | 20 | 7.52 | 2081 | 14.32 | 114 | 4.20 | 31 | 1.09 | 414 | 0.56 | 336 | 0.50 | 331 |
| **Average** | | **5.69** | **1698** | **18.91** | **89** | **2.44** | **18** | **0.81** | **269** | **0.46** | **145** | **0.47** | **152** |

**Table 2. Average computational results, uncapacitated (best) instances**

| Heuristic | # | LPMAH | | MAH | | CAH1 | | CAH2 | | UPH1 | | UPH2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | % | Seconds | % | Seconds | % | Seconds | % | Seconds | % | Seconds | % | Seconds |
| $m$=200 | 2 | n/a | n/a | 6.81 | 332 | 6.07 | 30 | 1.96 | 900 | 0.72 | 543 | 1.31 | 476 |
| $m$=250 | 9 | n/a | n/a | 27.99 | 568 | 3.05 | 79 | 1.03 | 1152 | 2.06 | 589 | 1.90 | 873 |
| $m$=300 | 20 | n/a | n/a | 31.62 | 707 | 4.51 | 84 | 1.31 | 1403 | 1.19 | 734 | 0.99 | 782 |
| $m$=350 | 20 | n/a | n/a | 40.85 | 945 | 3.24 | 133 | 1.42 | 1526 | 0.95 | 864 | 0.67 | 964 |
| $n$=50 | 10 | n/a | n/a | 29.78 | 391 | 1.55 | 50 | 1.01 | 539 | 2.22 | 61 | 0.98 | 62 |
| $n$=100 | 13 | n/a | n/a | 34.01 | 686 | 3.28 | 58 | 1.60 | 864 | 0.23 | 229 | 0.83 | 262 |
| $n$=150 | 13 | n/a | n/a | 32.02 | 869 | 5.86 | 141 | 0.98 | 1847 | 1.32 | 624 | 0.88 | 800 |
| $n$=200 | 15 | n/a | n/a | 37.23 | 979 | 4.02 | 135 | 1.61 | 2009 | 1.37 | 1776 | 1.39 | 1952 |
| **Average** | | n/a | n/a | **33.62** | **761** | **3.81** | **100** | **1.33** | **1387** | **1.23** | **751** | **1.04** | **857** |

In the case of the CTPP, results presented in Tables 3 and 4 show that CPH yields average results within 2% of the optimum when $m \leq 200$. The global average on the 439 instances with a known optimal solution is equal to 0.94%. With the exception of LPMAH which performs very well on capacitated instances, these gaps are smaller than those produced with the other heuristics. Computation times are comparable to those of LPMAH, larger than those of MAH and CAH1, and much smaller than those of CAH2. On the more difficult instances for which the optimum is not known, the gap with respect with the best known solution is typically less than 1%. The size of the gap seems to be related to the value of $\lambda$, the average gap being 0.02% for $\lambda$=0.1 and 3.22% for $\lambda$=0.99. Here, CAH2 becomes more time consuming and does not fare as well as CAH1 when $m \geq 300$.

**Table 3. Average computational results, capacitated (optimal) instances**

| Heuristic | # | LPMAH | | MAH | | CAH1 | | CAH2 | | CPH | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | % | Seconds | % | Seconds | % | Seconds | % | Seconds | % | Seconds |
| $m$=50 | 140 | 0.38 | 7 | 3.26 | 1 | 1.42 | 5 | 0.49 | 12 | 0.41 | 10 |
| $m$=100 | 135 | 0.69 | 58 | 6.16 | 7 | 2.00 | 33 | 1.06 | 86 | 0.97 | 75 |
| $m$=150 | 119 | 0.60 | 173 | 7.73 | 22 | 2.55 | 113 | 1.81 | 361 | 1.39 | 225 |
| $m$=200 | 45 | 0.50 | 248 | 5.64 | 43 | 4.61 | 156 | 2.81 | 1465 | 1.32 | 510 |
| $n$=50 | 131 | 0.69 | 132 | 7.94 | 14 | 3.32 | 34 | 1.72 | 461 | 1.11 | 162 |
| $n$=100 | 116 | 0.57 | 61 | 5.90 | 13 | 1.83 | 65 | 1.27 | 281 | 0.92 | 154 |
| $n$=150 | 94 | 0.44 | 86 | 3.99 | 11 | 1.51 | 58 | 0.87 | 141 | 0.78 | 113 |
| $n$=200 | 98 | 0.43 | 82 | 3.67 | 13 | 1.96 | 83 | 1.02 | 161 | 0.89 | 118 |
| $\lambda$=0.1 | 70 | 0.01 | 18 | 0.16 | 4 | 0.16 | 81 | 0.10 | 190 | 0.15 | 26 |
| $\lambda$=0.5 | 68 | 0.01 | 19 | 0.48 | 7 | 0.50 | 96 | 0.32 | 393 | 0.48 | 48 |
| $\lambda$=0.7 | 64 | 0.07 | 25 | 1.31 | 10 | 1.65 | 71 | 0.77 | 387 | 0.76 | 129 |
| $\lambda$=0.8 | 69 | 0.32 | 44 | 2.48 | 22 | 3.02 | 72 | 1.57 | 526 | 0.99 | 290 |
| $\lambda$=0.9 | 64 | 1.28 | 99 | 6.43 | 30 | 5.24 | 49 | 3.62 | 262 | 1.88 | 329 |
| $\lambda$=0.95 | 42 | 2.09 | 86 | 13.99 | 9 | 3.09 | 11 | 1.95 | 54 | 1.13 | 85 |
| $\lambda$=0.99 | 62 | 0.69 | 377 | 18.76 | 7 | 2.51 | 5 | 0.88 | 27 | 1.36 | 51 |
| **Average** | | **0.55** | **92** | **5.61** | **13** | **2.23** | **58** | **1.26** | **278** | **0.94** | **139** |

## 6- CONCLUSIONS

We have developed, implemented and tested several heuristics for two versions of the *Traveling Purchaser Problem*, a hard combinatorial optimization problem arising in production and transportation planning. Our three perturbation heuristics tend to produce better quality

solution than alternative heuristics. One exception arises in the CTPP where CPH is surpassed by LPMAH, a constraint generation LP based heuristic requiring a heavier programming machinery. On the whole, our implementation of the Commodity Adding Heuristic does not perform as well as the perturbation heuristics. It seems that the main difficulty in the TPP is to determine which markets enter the solution and this can only be achieved by applying in succession several post-optimization procedures as we have done in UPH1, UPH2 and CPH, even if this translates into increased computation times. In other words, simple heuristic rules do not seem to be the answer for the TPP.

**Table 4. Average computational results, capacitated (best) instances**

| Heuristic | # | LPMAH | | MAH | | CAH1 | | CAH2 | | CPH | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | % | Seconds | % | Seconds | % | Seconds | % | Seconds | % | Seconds |
| $m$=100 | 5 | n/a | n/a | 8.80 | 11 | 5.24 | 22 | 3.55 | 91 | 1.68 | 111 |
| $m$=150 | 21 | n/a | n/a | 12.04 | 46 | 6.28 | 60 | 3.96 | 261 | 2.97 | 460 |
| $m$=200 | 95 | n/a | n/a | 6.24 | 69 | 2.32 | 288 | 1.28 | 1096 | 0.99 | 643 |
| $m$=250 | 140 | n/a | n/a | 7.09 | 119 | 1.85 | 460 | 1.10 | 2109 | 0.54 | 1101 |
| $m$=300 | 140 | n/a | n/a | 7.31 | 225 | 2.21 | 838 | 2.57 | 2669 | 0.60 | 1697 |
| $m$=350 | 140 | n/a | n/a | 6.61 | 377 | 1.95 | 1403 | 9.08 | 3031 | 0.79 | 2198 |
| $n$=50 | 114 | n/a | n/a | 10.05 | 149 | 2.89 | 330 | 5.31 | 2341 | 0.78 | 1180 |
| $n$=100 | 129 | n/a | n/a | 7.56 | 176 | 2.37 | 581 | 5.72 | 2314 | 0.99 | 1358 |
| $n$=150 | 151 | n/a | n/a | 6.10 | 202 | 2.04 | 882 | 3.08 | 2097 | 0.68 | 1440 |
| $n$=200 | 147 | n/a | n/a | 5.36 | 261 | 1.89 | 1156 | 1.36 | 2185 | 0.80 | 1657 |
| $\lambda$=0.1 | 70 | n/a | n/a | 0.02 | 27 | 0.02 | 989 | 0.01 | 1838 | 0.02 | 720 |
| $\lambda$=0.5 | 72 | n/a | n/a | 0.06 | 74 | 0.07 | 1273 | 4.49 | 2726 | 0.05 | 724 |
| $\lambda$=0.7 | 76 | n/a | n/a | 0.21 | 124 | 0.34 | 1019 | 6.71 | 3072 | 0.09 | 1020 |
| $\lambda$=0.8 | 71 | n/a | n/a | 0.89 | 205 | 1.40 | 972 | 6.69 | 3266 | 0.10 | 1882 |
| $\lambda$=0.9 | 76 | n/a | n/a | 3.81 | 508 | 4.44 | 836 | 4.11 | 2979 | 0.22 | 2888 |
| $\lambda$=0.95 | 98 | n/a | n/a | 9.44 | 316 | 4.83 | 315 | 2.93 | 1630 | 1.51 | 1846 |
| $\lambda$=0.99 | 78 | n/a | n/a | 32.46 | 100 | 3.56 | 65 | 1.28 | 346 | 3.22 | 726 |
| **Average** | | n/a | n/a | **7.08** | **201** | **2.26** | **752** | **3.71** | **2224** | **0.80** | **1425** |

***Acknowledgements***

This research work was partially supported by grants OPG0036509, OPG0039682 and OPG0172633 from the Canadian National Science and Engineering Research Council. This support is gratefully acknowledged.

## REFERENCES

Basart, J. M. and L. Huguet (1989). An approximate algorithm for the TSP, *Information Processing Letters*, **31**, 77-81.

Buzacott, J. A. and S. K. Dutta (1971). Sequencing many jobs on a multipurpose facility, *Naval Research Logistics Quarterly*, **18**, 75-82.

Dowsland K. (1996). Genetic algorithms – a tool for OR?, *Journal of the Operational Research Society*, **47**, 550-561.

Golden, B. L., L. Levy and R. Dahl (1981). Two generalizations of the traveling salesman problem, *Omega*, **9**, 439-445.

Laporte, G., J. Riera-Ledesma and J.-J. Salazar-González (2000). A branch-and-cut algorithm for the undirected traveling purchaser problem, unpublished paper.

Ong, H. L. (1982) Approximate algorithms for the travelling purchaser problem. *Operations Research Letters*, **1**, 201-205.

Pearn, W. L. and R. C. Chien (1998). Improved solutions for the traveling purchaser problem, *Computers & Operations Research*, **25**, 879-885.

Ramesh, T. (1981). Traveling purchaser problem, *Opsearch*, **18**, 78-91.

Renaud J. and F. F. Boctor (1998). An efficient composite heuristic for the symmetric generalized traveling salesman problem, *European Journal of Operational Research*, **108**, 571-584.

Renaud J., F. F. Boctor and G. Laporte (1996). A fast composite heuristic for the symmetric traveling salesman problem, *INFORMS Journal on Computing*, **8**, 134-143.

Renaud J., F. F. Boctor and G. Laporte (2000). Perturbation heuristics for the pickup and delivery traveling salesman problem. *Computers & Operations Research*, forthcoming.

Singh, K. N., and D. L. Van Oudheusden (1997). A branch and bound algorithm for the traveling purchaser problem, *European Journal of Operational Research*, **97**, 571-579.

Voβ, S. (1996). Dynamic tabu search strategies for the traveling purchaser problem, *Annals of Operations Research*, **63**, 253-275.